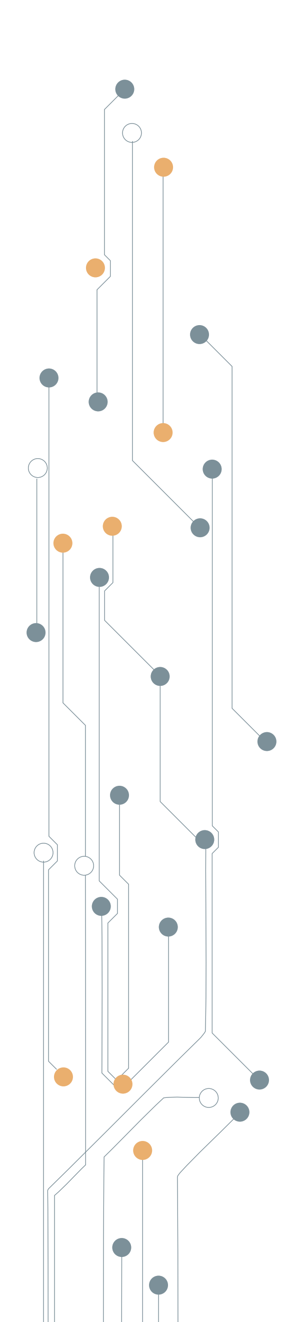
Arrays



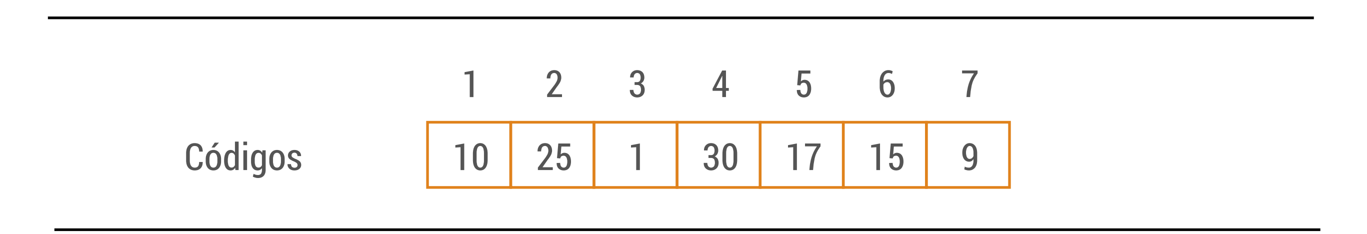
Índice



Arrays

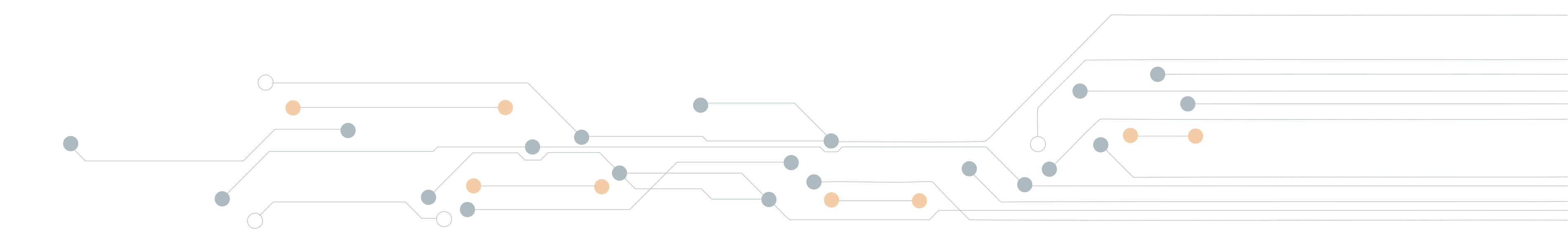
1. Arrays

Un array, también llamado tabla o matriz, es un conjunto de datos de un mismo tipo al que se accede mediante una única variable. Cada dato tiene una posición en el array:



En el ejemplo de la figura anterior, tenemos un array de 7 elementos de tipo número entero, contenido en la variable codigos. En este caso, la posición asignada al primer elemento es la 1, a siguiente la 2 y así sucesivamente.

Los arrays los utilizamos cuando queremos manejar un grupo de datos de manera conjunta, realizando una serie de operaciones sobre dichos datos. Al acceder a todos ellos mediante la misma variable y tener un índice asociado, el array se puede recorrer mediante una instrucción de tipo repetitivo, con lo que podemos definir las operaciones dentro de la estructura repetitiva y que estas se apliquen para cada uno de los elementos del array.



1.1 | Definición de un array

A la hora de declarar una variable de tipo array, lo haremos como con cualquier otra variable, solo que debemos indicar el rango de índices para establecer el tamaño del array:

```
codigos 1 to 7 Integer
```

En la instrucción anterior se ha declarado la variable códigos del ejemplo, consistente en un array de enteros de 7 elementos

Hay lenguajes de programación en los que el índice del primer elemento está fijado en 0 y no se puede modificar, por lo que a la hora de declarar el array simplemente se indicará el tamaño del mismo.

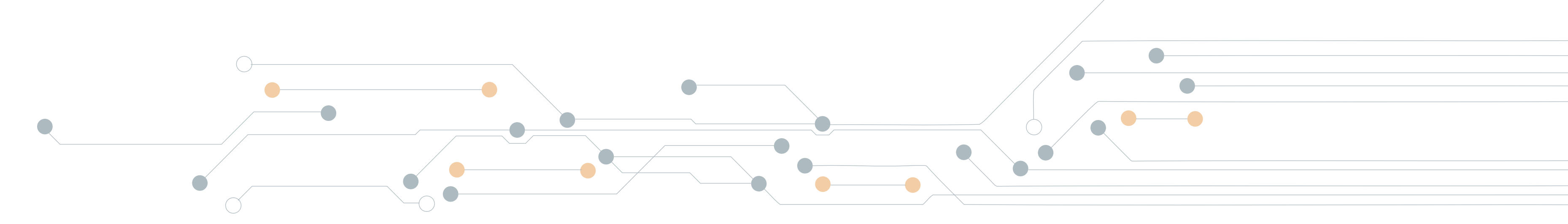
1.2 | Acceso a los elementos del array

Para poder acceder a una posición del array utilizaremos la expresión variable[indice]. Como vemos, después del nombre de la variable se indica entre corchetes la posición a la que queremos acceder. Por ejemplo, para almacenar un valor en la primera posición del array códigos sería:

```
codigos[1]=100
```

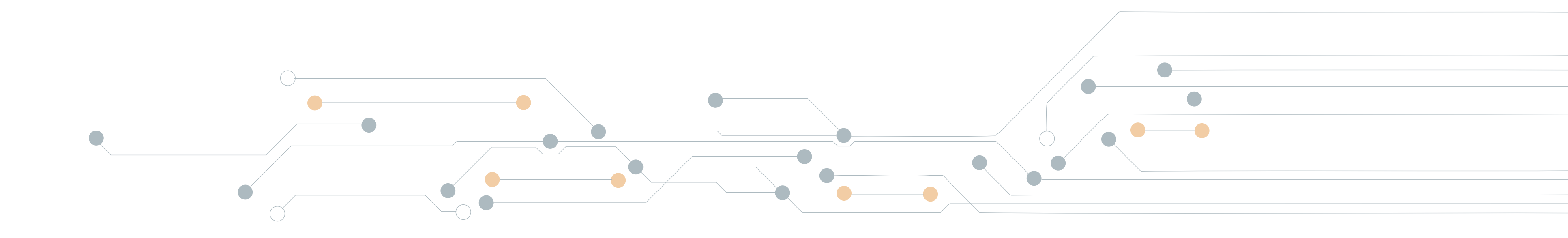
Y para decirle al programa que nos muestre el valor de la última posición sería:

```
Imprimir codigos[7]
```



El siguiente programa de ejemplo almacena los 10 primeros números pares en un array de enteros y después lo recorre para mostrar su contenido:

```
Inicio
      Datos:
      nums 1 to 10 Integer
      i Integer
      Código:
      For i=1 To 10 Step 1
            nums[i]=(i-1)*2
      End For
      For i=1 To 10 Step 1
            Imprimir nums[i]
      End For
Fin
```



1.3 | Arrays multidimensionales

El tipo de array que hemos tratado hasta ahora es un array de una dimensión, pero podemos tener arrays de varias dimensiones. Un array de dos dimensiones, por ejemplo, sería como una tabla organizada en filas y columnas:

	1	2	3	4	5	6	7
1							
2							
3							
4							

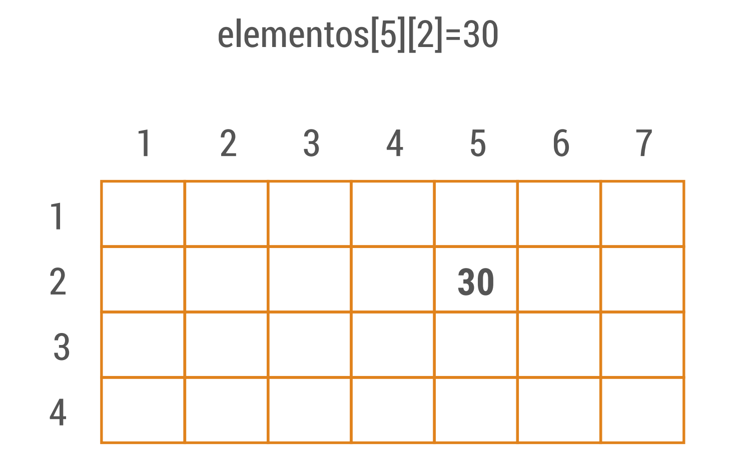
El array de la imagen anterior sería de dos dimensiones y tendría un tamaño de 4 x 7 = 28 elementos. Para declarar una variable de este tipo de array utilizaremos la expresión:

```
elementos 1 to 7, 1 to 4 Integer
```

La variable la hemos llamado *elementos* y se deben indicar los rangos de índices de cada dimensión, separados por una coma.



Para acceder a cada posición, utilizaremos doble corchete, uno para cada índice. Por ejemplo, para almacenar un número en la quinta columna y segunda fila sería:



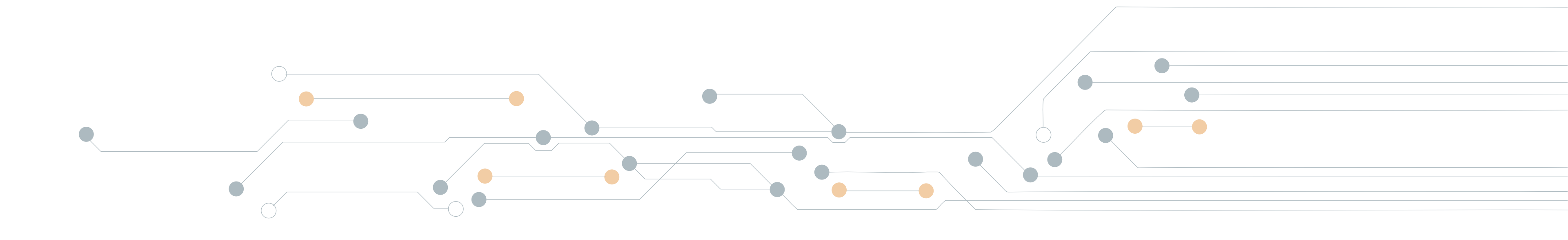
Para arrays con más de dos dimensiones el proceso sería el mismo, aunque a partir de tres dimensiones no podríamos hacer una representación gráfica del mismo.

En el siguiente ejemplo declaramos un array de tres dimensiones de tamaño 4x7x3:

```
datos 1 to 4, 1 to 7, 1 to 3 Integer
datos[2][5][1]=100
```

Para recorrer un array multidimensional, se deberán utilizar estructuras repetitivas anidadas, una por cada dimensión. El siguiente algoritmo de ejemplo simula la generación de un tablero de ajedrez mediante un array bidimensional, representando los cuadrados negros con 1 y los blancos con 0:

```
Inicio
        Datos:
        Tablero 1 to 8, 1 to 8 Integer
        I,K Integer
        For I=1 To 8 Step 1
                For K=1 To 8 Step 1
                        If((I+K)%2==0) Then
                                Tablero[I][K]=0
                        Else
                                Tablero[I][K]=1
                        End If
                End For
        End For
Fin
```



1.4 | Ordenación de arrays

Una de las tareas más habituales que se llevan a cabo con arrays es la ordenación de los mismos. Ordenar un array consiste en colocar sus elementos siguiendo un determinado criterio de ordenación, por ejemplo, en un array de números este criterio podría consistir en colocar los elementos de menor a mayor o de mayor a menor.

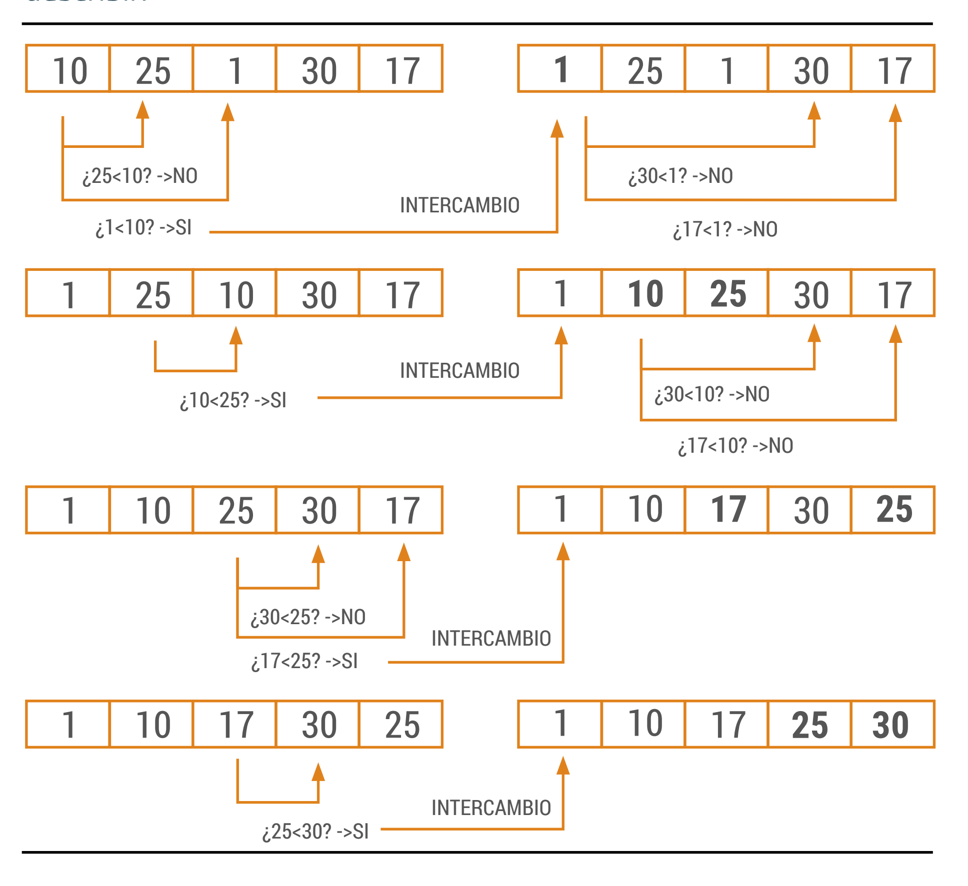
En cualquier caso, para conseguir ordenar un array debemos seguir algún método que nos permita definir un algoritmo que garantice dicha ordenación. Uno de esos métodos es el conocido como **método de la burbuja.**

El método de la burbuja consiste en ir realizando ordenaciones parciales de cada elemento del array, de manera que al final conseguiremos tener ordenados todos los elementos.

Más concretamente, el procedimiento a seguir para aplicar este método en el caso de un array numérico que deba ser ordenado de menor a mayor, consistirá en lo siguiente: para cada elemento del array, se comprueba si alguno de los siguientes elementos es menor que él, si es así realizamos el intercambio de los elementos y seguimos con las comprobaciones, si encontramos un nuevo elemento menor, volvemos a intercambiar, y así hasta llegar al último.

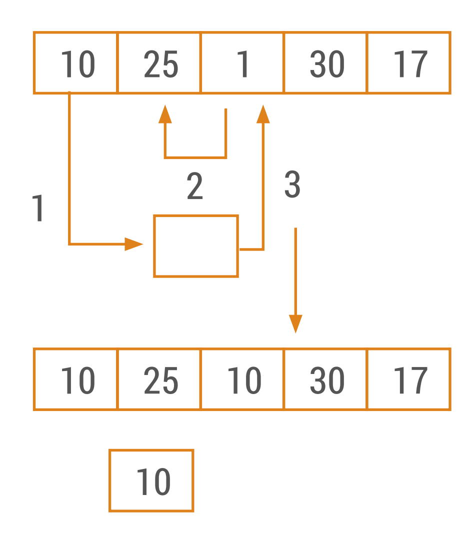
Si el proceso anterior se realiza con todos los elementos del array, excepto el último, se garantiza que el array quedará ordenado al finalizar todas las comprobaciones.

En la siguiente imagen se ilustra gráficamente el proceso de ordenación de un array de ejemplo, formado por los números 10, 25, 1, 30 y 17, utilizando el método de la burbuja que se acaba de describir:



La primera fila representa las comprobaciones del primer elemento del array, en la segunda fila se indican las comprobaciones con el segundo elemento, y así sucesivamente. Cuando se cumple la condición, se realiza el intercambio de los elementos y se continúa comprobando con los siguientes.

De cara a aplicar mediante pseudocódigo el método de la burbuja para ordenar un array, debemos utilizar dos instrucciones *for* anidadas, la primera para recorrer cada uno de los elementos del array y la segunda para realizar las comprobaciones entre cada elemento y los siguientes. Así mismo, se necesitará una variable auxiliar para poder realizar el intercambio de las posiciones, tal y como se ilustra en la siguiente imagen:



Para poder intercambiar el contenido de dos posiciones del array, primero se salva el contenido de una de las posiciones en la variable auxiliar, después se vuelca en dicha posición el contenido de la segunda y en tercer lugar, el valor de la variable auxiliar se lleva a la posición que fue volcada en la primera.

El siguiente algoritmo realiza la ordenación del array de notas que utilizamos en el ejercicio de ejemplo presentado al principio del apartado, utilizando la variable *aux* para realizar el intercambio de los datos:

```
For I=1 To 15 Step 1
      For K=I+1 To 15 Step 1
            If(notas[K]<notas[I]) Then
                  //realización del intercambio
                  aux=notas[I]
                  notas[I]=notas[K]
                  notas[K]=aux
            End If
      End For
End For
```


Telefónica
EDUCACIÓN DIGITAL